Background

I have been creating and maintaining software for over 30 years. Several years ago I was contracted to port Torchat to a derivative program labeled Onionchat (not released). In the course of that development project, several of the flaws present in the Torchat program were identified and corrected. There was an extensive list of future enhancements for Onionchat produced as the project neared a release candidate. Among the Torchat flaws identified and corrected, were a serious security issue (discussed below) and numerous corrections to error handling and error recovery. Only the original developers of Torchat have more familiarity with the Torchat source code and functionality.

The problems with Torchat chat logs as evidence.

Torchat implemented a simple logging function that stored the chat message text in a plain text file. Each message was written to that text file with a timestamp as a message was sent or received by the Torchat program with logging enabled (logging is not enabled by default).

There was no effort made by the original Torchat developer to ensure the log file was in any way secure. A log file intended to be used as evidence should have the following features: 1) it should be tamper-proof, 2) it should not be possible for any third party to create (counterfeit) such a log file.

Proper Torchat Logging was Possible

The Torchat program had all the information and digital security keys needed to meet those requirements for a secure, inviolable chat log file. The Tor network enforces a time synchronization to even permit a connection (typically less than 60 seconds). If the timestamp of either party is outside the permitted limit, the Tor network will not permit the connection or will terminate an existing connection. The Tor message packets are encrypted for transmission over the Tor network. It would not have been difficult for the sending Torchat program to format the message with timestamp, digitally sign it with the private security key, and then transmit the message over the Tor network. This would provide logged Torchat messages that were essentially tamper-proof and nearly immune to counterfeiting or modification as the timestamps are part of the digitally signed messages.

Torchat and Text Files

What the Torchat program did when logging was turned on, was to format the message with a generated timestamp and write that line of plain text into a simple text file. In this simple file format, the only valid way to be certain that the Torchat log file had not been tampered with would be have log files of both sides of a Torchat conversation and verify the chat text matched in the log files of both parties to the Torchat sessions. (This only works if one of those log files has a secure chain of custody during the generation of the file, as it is possible to fabricate Torchat log files for both sides of a fictitious Torchat conversation.)

Text files are nearly the simplest file format in use. Lines of text are terminated with carriage-return and optionally linefeed. (typewriter terminology) The only forensic data provided with a text file is the time of the last modification. That modification could be a single character change, or the entire contents of the file added, deleted or re-written. There is no way to ever know what changed. Sadly this means that any clever hacker could change anything (or everything) in a text file, write the results, and then restore the file timestamp to any desired time without detection.

Torchat and Impersonation

There is also the matter of proving who, exactly, is on the other end of a Torchat session. Torchat was intended to provide a means of communicating privately with no possibility of the messages being intercepted in transit and no way to trace the sender or receiver. This opens the possibility for the remote chat participant to impersonate someone else with minimal risk of detection.

Presumably, the Torchat user who has possession of the Torchat log file also logged the conversation and thus is one of the participants in those logged chat conversations. Unfortunately this is not absolutely guaranteed due to an extremely serious security flaw in the commonly available versions of the Torchat program V0.9.x (details of this flaw discussed later).

Authorship Analysis (Stylometry or Wordprint)

The anonymity of Torchat and the Tor network makes identification of the individual at the remote end of the conversation extremely difficult. One possible means of identifying who was the remote user in a Torchat log file, is to perform what is called an authorship analysis using stylometry or wordprint techniques.

Stylometry and Wordprint programs (and experts) will examine features such as word use, sentence structure, sentence complexity, vocabulary, use of word case, hyphenation, punctuation, use of passive voice, common phrases, abbreviation usage, slang usage, miss-spellings, and inaccurate verb forms to provide a path to identifying authorship.

Because they can be hard to control, the analysis of errors and other ungrammatical elements that may be unique to the author, such as incorrect spellings, misuse of words and inaccurate verb forms, have achieved reasonably high accuracy in author identification when combined with other features.

While authorship analysis will indicate whether an individual suspected of being the remote participant is likely to be the author of the chat messages or is unlikely to be the author, authorship analysis generally cannot provide definitive proof of authorship.

The Security Flaw in Torchat V0.9.x

The original Torchat program was written by a college student in Germany. Generally the code was well done and most of the flaws that exist can be attributed to inexperience. The most serious flaw was an inherent security design flaw in the file sharing feature.

Torchat allowed users to transfer and share files with their remote chat participant. The file transfer feature was fairly robust and worked with any kind of file. There was a buffer retry error, (fixed in the OnionChat derivative of Torchat) yet that error only affected very large files, causing a timeout after more than 10 transmission errors. Files under a megabyte in size rarely failed to transfer.

The Torchat approach taken to transmit a file, was that the local user would select the file transfer dialog, browse and select the desired file, then click “send” to transfer that file to the remote user. The file transfer was started immediately without the permission of the remote user.

Torchat V0.9.x did not require the remote user to accept the file, an extremely serious security failure. When the local user clicks “send”, the file transfer is started immediately. The remote user’s Torchat application writes the file contents into a temporary file while the transfer is in progress, and upon successful file transfer, renames the file and presents a dialog asking the remote user where to put the file. The default location for receipt of transferred files is the exact same location that chat log files are created and stored. An inattentive remote user could click <OK> on the file transfer dialog without realizing it was a file transfer dialog instead of the more frequent connect / disconnect notices.

This behavior opens the door for malicious actors to indulge in serious mischief. Potential abuses include transferring various forms of illegal pornography, or indeed even fabricated chat log files. All the malicious actor has to do is start a file transfer while the remote user is away or inattentive and wait for the transfer to complete then disconnect the chat session. A really malicious user may connect and disconnect numerous times to create a series of dialog boxes on the remote end stating “connection lost”. A file save dialog is easily passed by as an inattentive user clicks on the informational dialog boxes to get rid of the screen clutter. The remote user’s save file dialog may close, or may stay open until the window is touched with a mouse cursor, yet it will disappear. This leaves the transferred file on the remote user’s computer, in the chat log directory, without the remote user’s permission and possibly without the remote user’s knowledge that a file was deposited on his computer.

How these security weaknesses and flaws in the Torchat log files can potentially be exploited is illustrated in the following 2 fictional scenarios.

Fictional Scenario One, an altered chat file as an alibi.

In this scenario, imagine the county party chairman chatting with an active member of a local political action group (PagMember) on a regular basis. The use of Torchat, was sold to the county party chairman on the basis of anonymity. Both parties generally keep the Torchat application running to send messages and informally debate various political issues at lunchtime once or twice a week.

PagMember has enabled logging in Torchat and has been regularly logging the chat sessions. PagMember has also observed that if he and the county party chairman debate on a Monday or Tuesday, the county party chairman won’t converse over Torchat again until maybe Thursday or Friday.

After a lengthy Monday Torchat debate, PagMember plans a bank robbery Tuesday at lunchtime (12:20). PagMember arranges to have all the surveillance camera lenses in the surrounding blocks spray-painted Monday night for a protest. Masked & hooded, the robbery is executed successfully Tuesday at lunchtime.

Immediately after the robbery, PagMember opens his Torchat log file with notepad (standard Windows program) and alters all the timestamps on the Monday chat session to overlap the Tuesday bank robbery. He advances the timestamps by a day and 10 minutes in every one of Monday’s discussion timestamps to overlap the entire robbery time frame and saves the text file. Then PagMember stops and restarts Torchat to disconnect & reconnect with the county party chairman’s Torchat to update the timestamp on his log file.

When investigators come calling a couple days later, PagMember denies all knowledge of the bank robbery and eventually tells the investigators he has an alibi for the time of the robbery, telling them he was chatting with the county party chairman at the time of the robbery.

PagMember provides the investigators with a copy of his Torchat log file and when the county party chairman is questioned, he remembers the debate clearly. The county party chairman is off by a day, yet he clearly remembers the chat conversation and the contents of the log file confirm the county party chairman's recollection of the conversation.

Without any other evidence, PagMember is no longer a prime suspect.

There is no way to detect or prove that the contents of a text file created as a Torchat log file has been altered, or what has been altered.

Fictional Scenario Two, a chat file planted as evidence.

In this scenario, a local lawyer has been asked to encourage a neighbors nephew (N-nephew) in his studies at university. N-nephew is struggling with the coursework and the lawyer has reluctantly agreed to occasionally chat with N-nephew via Torchat about homework issues.

The arrangement has worked out to be just the occasional evening chat session, as unknown to the lawyer, N-nephew has already decided to pursue a different course of study or quit at the end of the current semester. Also unknown to our lawyer, N-nephew is logging the Torchat conversations.

Our lawyer happens to walk past the bank being robbed in Scenario One as it was happening, and casually remarks to N-nephew the next evening that he didn't notice a thing even though according to reports the robbery was in progress at the exact time he walked past. Shortly after, they say goodnight.

Intrigued, N-nephew searches for details of the robbery and finds numerous Facebook postings of eyewitnesses with many details not present in the news reports.

When the local police announce a $25k award for information leading to the arrest and conviction of the bank robber, N-nephew hatches a plan to collect the reward (tax free as the IRS does not tax informant monies).

N-nephew alters the chat log, creating a fictional narrative about committing the bank robbery complete with many of the extra Facebook details (woman fainting, chrome pistol, etc). Then N-nephew does a couple of massive search and replace operations, replacing the "myself" identifier with his own Torchat address (16 characters) and replacing the lawyer's Torchat address with "myself" to make the log file appear to be the lawyer's Torchat log file.

Next, N-nephew renames the file with his own Torchat address (address.log) and transfers the file to the lawyer's computer and disconnects upon completion of the file transfer. This leaves the log file on the lawyer's computer with a timestamp matching the latest disconnect. N-nephew connects and disconnects numerous times to leave a clutter of connection dialog boxes on the remote computer and obfuscate the file save dialog box. N-nephew does not re-connect after that.

Next, N-nephew calls the tip line with information that the lawyer had bragged of the crime to him over a chat session.

Investigators don't believe the tip, yet surveillance cameras in the area show the lawyer entering the area and leaving the area at the appropriate time, so they reluctantly focus on the lawyer as he is now

their only suspect. They get a warrant to search his home, and encounter strenuous resistance over any examination of the lawyer's home computer.

After much wrangling about the warrant, confidential client files and the scope of examination, the computer is eventually examined. Discovery of the Torchat log file is cause for great celebration for the investigators. Details such as the eyewitness testimony about the actual robber being perhaps 3 inches shorter and miss-matches in clothing and shoe descriptions are glossed over.

The Torchat log file is devastating as the lawyer cannot explain how it got there nor how it was filled with all the unpublished and devastatingly accurate details of the bank robbery. Indictment & ruination quickly follow.

Post-conviction and informant money payout, N-nephew takes a 3 month European vacation with his tax-free $25k and returns to university to pursue a Masters degree in political science.

Conclusion

The above scenarios are entirely fictional, yet illustrate the potential problems with trusting Torchat log files as evidence.

Plain text (no digital signature) stored in a simple text file is far too easily manipulated without a trace to be considered reliable evidence without a rigorous chain of custody from initial creation, through to the ultimate presentation as evidence.

Without a rigorous chain of custody, a Torchat log file should only be considered valid and reliable for evidence when the log files of both sides of a Torchat conversation are verified to match the chat text of both parties to the Torchat sessions, with one file generated under a rigorous chain of custody. It is entirely feasible to fabricate either or both sides of an imaginary Torchat conversation or to fabricate the remote side of a Torchat conversation from a local Torchat log file.

___/s/_____
Michael Weigand

March 21, 2021